Fast and Accurate Wire Delay Estimation for Physical Synthesis of Large ASICs

Ruchir Puri David S. Kung
ruchir@us.ibm.com, kung@us.ibm.com
IBM Thomas J. Watson Research Center
Yorktown Heights, NY - 10598

Anthony D. Drumm
drumm@us.ibm.com
IBM Corporation
Rochester, MN - 55901

Categories and Subject Descriptors

B.7 [**Hardware**]: Integrated Circuits

General Terms

Algorithms, Design

Keywords

Integrated Circuit Design, Wire Delay, Estimation, Placement Driven Synthesis

ABSTRACT

Interconnect delays represent an increasingly dominant portion of overall circuit delays. During timing-driven physical synthesis process, timing analysis is repeatedly performed over several hundred thousand components. Thus, fast and accurate estimation of interconnect delays is crucial. Traditionally, lumped and elmore delay models have been widely used for computing interconnect delays in physical synthesis due to their computational efficiency. However, these delay models are known to be inaccurate since they ignore slew and resistive shielding effects. In this paper, we propose a new iterative refinement based delay estimation approach that considers resistive shielding along with driver slew. Experimental results show that the proposed approach gives not only highly accurate results for far end RC-line delays but also compares very favorably to more difficult to match source end delays and source slews. In addition, use of the proposed delay model in physical synthesis yields significant performance improvement on several large industrial ASICs.

1. INTRODUCTION

At higher levels of design, e.g., in physical synthesis applications, the physical location as well as size and number of various circuits is constantly changing due to various logical and physical optimizations. Thus, the interconnect topology of various nets in the design are also being changed as the placement and synthesis interaction progresses. These changing net topologies require several hundred thousand evaluations of interconnect delays during each iteration of placement and synthesis interaction. In addition, even if interconnect topology of a net does not change, a change in source slew changes the interconnect delays due to resistive shielding which is slew dependent.

Past efforts have employed either simple but computationally efficient lumped RC delay model and elmore delay model for fast computation of interconnect delays; or higher-order moments based delay metrics which are more accurate but prohibitively expensive to be used in a placement driven synthesis application. In general, various interconnect delay estimation techniques operate on an RC-tree extraction of the interconnect structure, where every node has a capacitance to ground, i.e., there are no floating capacitances and adjacent nodes are connected by resistors. Lumped RC delay model is a highly simplified estimation metric of the interconnect delay [2]. It is well known that the response of a simple RC circuit to a step function is given as $V(t) = V_0(1 - e^{-t/RC})$. This implies that the step input delay through the 50% point ($\frac{V_0}{2}$) of the output waveform for a simple RC circuit is given as $0.69 * RC$. For example, for the RC-tree shown in Figure 1, total resistance from source node 0 to node 7 is $R_1 + R_2 + R_4 + R_7$ and the total lumped net capacitance is $C_{total} = (C_0 + C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7)$. Thus the interconnect delay from source node 0 to node 7 according to lumped RC delay metric is given as $0.69 * (R_1 + R_2 + R_4 + R_7) * C_{total}$.

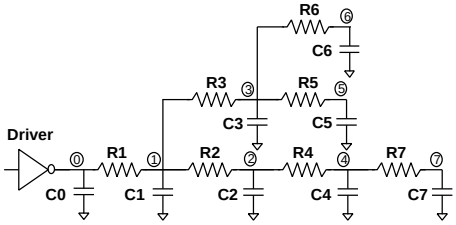

Figure 1: RC-tree representation of an Interconnect

Although lumped model is computationally very efficient, it can be highly inaccurate w.r.t real interconnect delays. Computing the exact delays of interconnect RC trees is expensive since it requires the exact solution of a set of dif-

GLSVLSI'02, April 18-19, 2002, New York, New York, USA.

ferential equations for various RC sections of the tree. One of the most popular delay metrics for RC trees has been the elmore delay because of its simplicity and reasonable correlation to real delays. Elmore originally estimated the 50% delay of a monotonic step response by the mean of the impulse response [5]. Rubinstein et al. [13] proved that response of a general RC tree to a step input is monotonic. They utilized elmore delay for obtaining bounds on the step response waveform of an RC tree. Under elmore delay, the signal delay T_{0-i} from source node 0 to some node i in the interconnect RC tree is given as:

$$T_{0-i} = \sum_{over\ all\ nodes\ k} R_{k,i} C_{tot_k} \quad (1)$$

where, C_{tot_k} represents the total downstream capacitance at node k; and $R_{k,i}$ represents the summation of all the resistances that are common between: the path from source node 0 to node i, and the path from source node 0 to node k. For example, in the interconnect RC tree in Figure 1, $R_{6,3}$ is given by $R_1 + R_3$. It is well known that computation of interconnect delays using elmore approach is very fast since it relies on single topological traversal of the interconnect tree. However, the elmore model can result in significant errors w.r.t real delays, especially in the case of deep-submicron designs where resistive shielding have significant effect on the interconnect delays. For example, elmore delay from source node 0 to node 2 in RC-tree of Figure 1 is given by $R_1(C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7) + R_2(C_2 + C_4 + C_7)$. In the limiting case when $R_3 = \infty$, elmore delay does not reflect the fact the downstream capacitances C_3, C_5, and C_6 are totally shielded from resistance R_1. This can cause significant error in node delays for nodes closer to the source on long nets. Since elmore delay represents the first moment of the impulse response, utilizing higher order moments can yield much better delay accuracy [6][7][9][14] . Unfortunately, most of these techniques utilizing higher order moments are computationally very intensive and require large run-times. Due to high computational complexity of obtaining higher order moments and a lack of an accurate but efficient delay metric, elmore delay has remained a popular metrics.

Recently, Kashyap et al. [8] proposed an interconnect delay metric that takes resistive shielding into account without computing higher-order moments. It relies on effective capacitance calculation [11][12] to account for resistive shielding. Kashyap et al. extended their effective capacitance metric by approximating a ramp input applied to an RC circuit with a step input applied at the instant when the ramp crosses the 50% point [8]. However, in practice, interconnects are driven by CMOS circuits (as shown in Figure 1) that drive the RC-interconnect wires with finite slew waveforms. The slew of this waveform is dependent on the effective capacitance load seen by the CMOS driver. Unfortunately, Kashyap's method for calculating effective capacitance relies on the slew value applied to the source of RC-tree which in turn depends on the effective capacitance itself in the case of CMOS drivers. Due to this drawback, it is difficult to apply Kashyap's method to compute accurate interconnect delays in CMOS logic circuits. In this paper, we propose a new computationally efficient and accurate wire delay estimation method that overcomes this drawback in Kashyap's method. In addition, we directly account for the finite slew of the ramp waveform in our delay metric as opposed to approximating it with a time shifted step input in Kashyap et al.'s method.

Our new estimation method [10] considers the effect of slew as well as resistive shielding of capacitance to yield more accurate delays for both interconnects and driver gate. To account for interdependence of slew and effective capacitance, we iteratively refine the source slew to yield accurate node delays. In addition, it is computationally similar in efficiency to the elmore delay model.

The rest of the paper is organized as follows. In Section 2.1, we discuss the effect of ramp inputs and resistive shielding in computing accurate effective load capacitance. Section 2.2 discusses our iterative refinement approach for computing interconnect delays. A detailed discussion of slew propagation mechanism used in the proposed iterative refinement approach is given in section 2.3. Section 2.4 illustrates the proposed approach with a simple example. Experimental results are given in section 3 and finally secion 4 concludes this paper.

2. INTERCONNECT DELAY ESTIMATION

2.1 Computing Effective Capacitance with ramp inputs

The effective capacitance seen at each node of a RC tree depends on the resistive shielding which in turn depends on the node slew, its resistance, and capacitance. In this section, we derive this dependence of the effective capacitance. In contrast to [8], where the effective capacitance is calculated for a step input and then approximated for ramp inputs, we consider the effect of ramp inputs on effective capacitance explicitly.

Let us consider a simple voltage source $V(t)$ driving a π-model RC-load $C_1 - R - C_2$ as shown in Figure 2. Let $I(t)$ be the total current provided by voltage source $V(t)$ and let $I_1(t)$ be the current through C_1 and $I_2(t)$ be the current through $R-C_2$. In the following, we perform simple analysis in frequency domain of the π model.

$$I(s) = I_1(s) + I_2(s)$$

$$I_1(s) = \frac{V(s)}{1/C_1 s}, I_2(s) = \frac{V(s)}{R + 1/C_2 s}$$

$$I(s) = V(s)(C_1 s + \frac{C_2 s}{1 + RC_2 s})$$

We assume that the source $V(t)$ is a ramp voltage source with a rise time t_r as shown in Figure 2. Thus, $V(t)$ is given by:

$$V(t) = \frac{V_{dd}}{t_r} t \ for\ t\ <\ t_r\ and\ V(t) = V_{dd}\ for\ t\ >=\ t_r$$

$$V(s) = \frac{V_{dd}}{t_r} \frac{1}{s^2}(1 - e^{-st_r})$$

Substituting this in the current equation above, we get:

$$I(s) = \frac{V_{dd}}{t_r}(\frac{C_1}{s} + \frac{C_2}{s(1 + RC_2 s)})(1 - e^{-st_r})$$

$$I(s) = \frac{V_{dd}}{t_r}(\frac{C_1 + C_2}{s} - \frac{C_2}{s + \frac{1}{RC_2}})(1 - e^{-st_r})$$

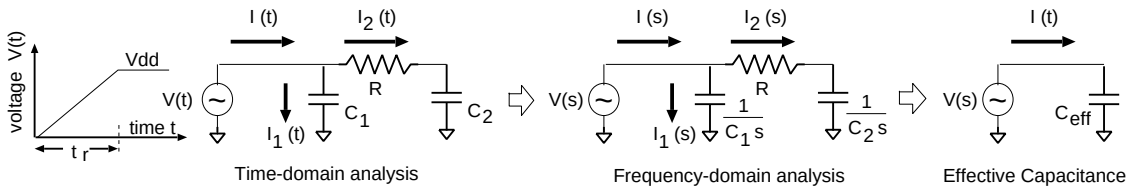

Figure 2: Calculation of Effective Capacitance with ramp inputs

$$I(t) = \frac{V_{dd}}{t_r}((C_1 + C_2) - C_2 e^{-\frac{t}{RC_2}}) \; for \; t \; < \; t_r$$

We are interested in calculating the effective capacitance as seen by the voltage source V(t). For this purpose, we define the effective capacitance C_{eff} to be a capacitance that requires the same charge transfer "Q" as that required by the π model load upto 50% delay point (i.e., the time when the input reaches $V_{DD}/2$, i.e., $t = t_r/2$). This charge "Q" is given by:

$$Q = \int_0^{t_r/2} I(t)dt = \int_0^{t_r/2} \frac{V_{dd}}{t_r}((C_1 + C_2) - C_2 e^{-\frac{t_r}{RC_2}})dt$$

Also, the charge transfer for charging the effective capacitance upto 50% delay point is given by $\frac{C_{eff}V_{dd}}{2}$. As discussed above, equating these two charge transfers, we obtain:

$$\frac{V_{dd}(C_1 + C_2)}{2} - \frac{RC_2{}^2 V_{dd}}{t_r}(1 - e^{-\frac{t_r}{2RC_2}}) = \frac{C_{eff}V_{dd}}{2}$$

$$C_{eff} = C_1 + C_2(1 - \frac{2RC_2}{t_r}(1 - e^{-\frac{t_r}{2RC_2}}))$$

Thus, $C_{eff} = C_1 + C_2 * K$, where K is the capacitance shielding factor defined as:

$$K = 1 - 2x(1 - e^{-\frac{1}{2x}}), \qquad where \qquad x = \frac{RC_2}{t_r}. \qquad (2)$$

Thus, capacitance shielding factor K depends on the time constant RC_2 and the input slew rate t_r. It is possible for the capacitance C_2 to be shielded by a significant amount depending on the relative values of RC_2 and slew rate t_r, as shown in Figure 3.

In practice, the voltage source V(t) driving the RC-tree is a CMOS gate whose input signal slew rate is known. Since we do not know the slew rate at the output of the gate (i.e., input of RC-tree), because it in turn depends on the effective capacitance, we solve for the value of C_{eff} with the driver slew equation. In standard cell based designs, delay and slew at output of a given cell is pre-characterized in timing rules in terms of its input slew and capacitive load. Since the input slew of the driver gate is known and fixed, the output slew at the driver can be expressed as a function of its output load. We utilize this interdependence of slew and capacitance and propose an iterative refinement method for determining interconnect delays. In this proposed method, we iteratively refine the delay estimates by repeated forward and backward traversal of the RC tree topology. During every iteration we refine the source slew, i.e., driver output slew in order to obtain successively accurate values of effective capacitance.

2.2 Iterative delay refinement approach

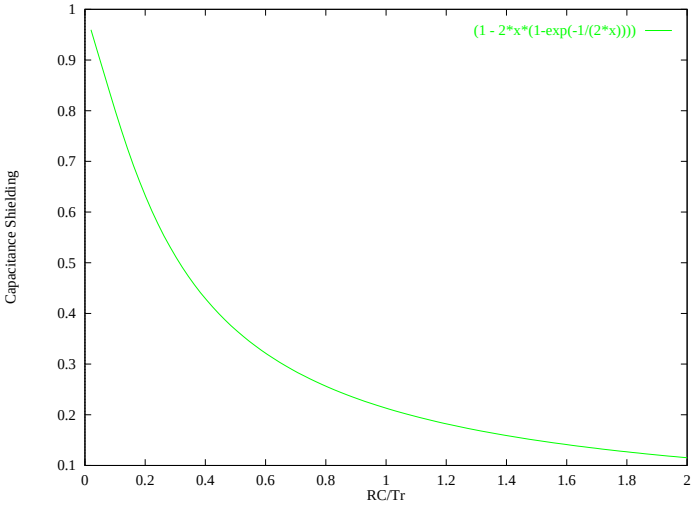

Figure 3: Variation of capacitance shielding factor K with ratio of RC-delay and input slew (RC_2/t_r).

We assume that the interconnect structure has been extracted in the form of an RC-tree (as shown in example Figure 1) where every node i has a capacitance C_i to ground, and has a resistance R_i that connects it to its parent node. Let, the source node 0 be driven by a CMOS driver gate whose output slew S_{driver} is modeled using timing equation : $S_{driver} = f$(gate capacitive load). We denote the total downstream capacitance at node i by C_{tot_i}; the slew at node i by S_i; and the effective capacitance at node i by C_{eff_i}.

The outline of our iterative refinement method for estimating interconnect delays is given as follows:

1. Initialize:
   - Effective capacitance C_{eff_i} of each RC tree node i with the sum of all downstream capacitances C_{tot_i}.
   - Slew at the source of RC-tree S_0, (i.e., driver output slew) from the driver slew equation with the driver load equal to the lumped net capacitance: $S_0 = f(C_{tot_0})$.
2. Traverse *forward* from source node towards sinks and compute :
   - Delay T_{0-i} from source 0 to each tree node i using elmore delay calculation (equation 1) with node's capacitance value equal to its effective ca-

pacitance, i.e.,

$$T_{0-i} = \sum_{over\ all\ nodes\ k} R_{k,i} C_{eff_i}$$

- Slew S_i at each tree node i using the parent slew and the tree segment delay (as discussed in section 2.3).

3. Traverse *backward* from source node towards sinks and compute :

   - Effective capacitance C_{eff_i} at each node with the summation of the node capacitance C_i and all the children nodes effective capacitances:

$$C_{eff_i} = C_i + \sum_{over\ all\ children\ nodes\ j} K_j * C_{tot_j}$$

   where shielding factor

$$K_j = 1 - \frac{2R_j C_{eff_j}}{S_i}(1 - e^{-\frac{S_i}{2R_j C_{eff_j}}})$$

4. Recompute the source slew S_0 using the driver slew equation with the gate capacitive load being the source node's effective capacitance C_{eff_0} from previous backward propagation pass, i.e., $S_0 = f(C_{eff_0})$.

5. If driver slew did not converge within a specified threshold of previous slew, iterate over step 2 to step 5 again until the slew at the driver output converges.

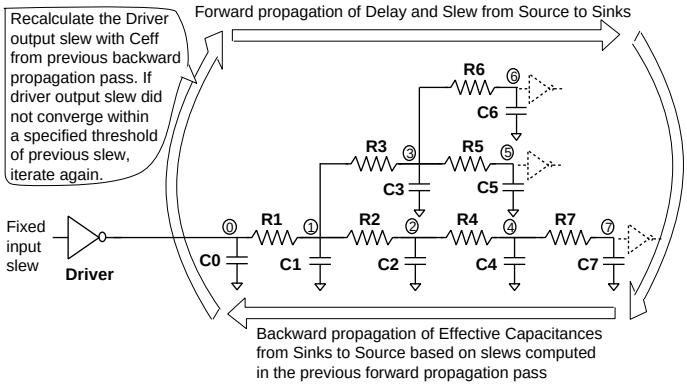

Figure 4: Outline of the proposed Iterative Refinement method for interconnect delay estimation.

The iterative refinement approach for estimating interconnect delays is also illustrated in Figure 4. As discussed above, the iterative refinement approach utilizes effective capacitance computations that explicitly considers effect of ramp inputs. In addition, we successively refine the node slew values during each pass of the tree topology to accurately compute the resistive shielding effect for each node. These resistive shielding factors are used to obtain highly accurate effective capacitance for each node. We use these effective capacitances in the elmore delay computations to obtain interconnect delays. Although we do not derive the proof of convergence of the iterative refinement approach here, experimental results with interconnects in all the large industrial ASICs, as well as randomly generated RC-lines show that the slew converges within 1% of the previous slew in less than 5 iterations.

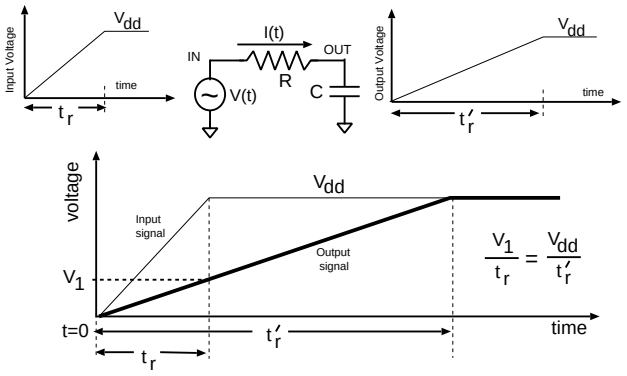

Figure 5: Slew Propagation in a simple RC-tree segment

2.3 Propagation of slew through an RC-tree

As discussed above, we propagate the slew from the source towards the sinks in order to obtain accurate capacitive shielding coefficients. In this section, we derive a relationship between the output slew and input slew of a simple RC segment. This relationship is used to propagate the slew in the RC-tree, as discussed in previous section.

Let a simple RC segment circuit as shown in Figure 5 be driven by a ramp input voltage source $V(t) = \frac{V_{dd}}{t_r}t$, where t_r is the slew of this ramp. The output voltage in this case can be straight forwardly derived as:

$$\frac{V_{dd}}{t_r}(t - RC_2 + RC_2 e^{-\frac{t}{RC_2}})$$

At t = t_r, the output voltage is given by V_1 (as shown in Figure 5):

$$\frac{V_{dd}}{t_r}(t_r - RC_2 + RC_2 e^{-\frac{t_r}{RC_2}})$$

Also from Figure 5, we can see that:

$$\frac{V_1}{t_r} = \frac{V_{dd}}{t'_r},$$

where t'_r is the slew of the output as shown in Figure 5. Thus, output slew t'_r is given by :

$$t'_r = \frac{V_{dd} t_r}{V_1}.$$

Substituting V_1 into this equation, we get:

$$t'_r = \frac{t_r^2}{t_r - RC_2 + RC_2 e^{-\frac{t_r}{RC_2}}}.$$

This equation can be simply reduced to:

$$t'_r = \frac{t_r}{1 - \frac{RC_2}{t_r}(1 - e^{-\frac{t_r}{RC_2}})} \quad (3)$$

Thus, given the R-C delay of the segment RC_2, and its input slew t_r, we can calculate the output slew t'_r with the above derived formula.

This relationship yields the dependence of output slew of a R-C segment on input slew as a function of ratio $x = \frac{RC_2}{t_r}$ as follows :

$$t'_r = \frac{t_r}{1 - x(1 - e^{-\frac{1}{x}})}$$

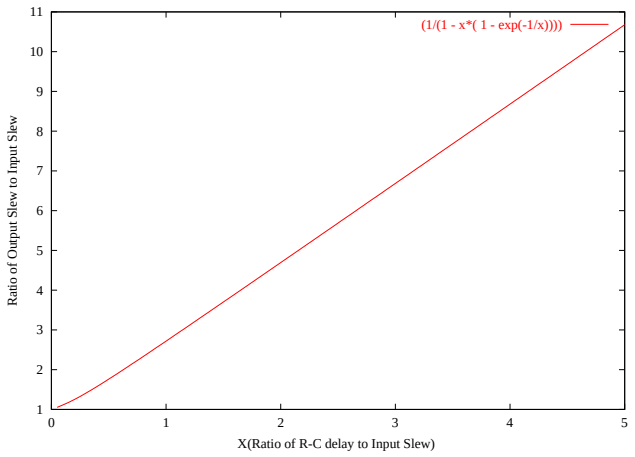

Figure 6: Dependence of output slew and input slew ratio (i.e., $\frac{t'_r}{t_r}$) on the ratio of RC-delay to input slew (RC_2/t_r).

Figure 6 shows the dependence of the ratio of output slew to input slew (i.e., $\frac{t'_r}{t_r}$) on the ratio of input slew to R-C delay (i.e., $\frac{RC_2}{t_r}$).

In the following section, we further illustrate our iterative refinement method on a simple example interconnect RC tree shown in Figure 1.

2.4 Example

Given a interconnect RC tree topology with a gate driver as shown in Figure 1, initially, we set the effective capacitance C_{eff_i} at every node i of RC tree to be the sum of all downstream capacitances C_{tot_i}. Thus, in the example tree, the effective capacitances at various nodes are initialized as:

$C_{eff_0} = C_0 + C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7 = C_{tot_0}$
$C_{eff_1} = C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7 = C_{tot_1}$
$C_{eff_2} = C_2 + C_4 + C_7 = C_{tot_2}$
$C_{eff_3} = C_3 + C_5 + C_6 = C_{tot_3}$
$C_{eff_4} = C_4 + C_7 = C_{tot_4}$
$C_{eff_5} = C_5 = C_{tot_5}$
$C_{eff_6} = C_6 = C_{tot_6}$
$C_{eff_7} = C_7 = C_{tot_7}$

Then we calculate the slew S_0 at the source node 0 by considering the driver slew equation ($S_{driver} = f$(gate capacitive load)) and substituting lumped source node capacitance C_{tot_0} for the gate capacitive load. Thus slew at source node S_0 is given by $f(C_{tot_0})$.

After determining the initial source node slew S_0, we traverse forward from source towards the sinks in a levelized manner (i.e., in order 0, 1, 2, 3, 4, 5, 6, 7) and determine node delays and slews as follows:

First we visit node 1 and determine the delay from the source: $T_{0-1} = R_1 C_{eff_1}$. Then we determine the slew at node 1 from the parent slew S_0 using the formula derived in equation 3 as :

$$S_1 = \frac{S_0}{1 - \frac{R_1 C_{eff_1}}{S_0}(1 - e^{-\frac{S_0}{R_1 C_{eff_1}}})}.$$

Then we compute the delay $T_{0-2} = R_1 C_{eff_1} + R_2 C_{eff_2}$ at node 2 and propagate node 1 slew S_1 to node 2 and determine its slew S_2 as:

$$S_2 = \frac{S_1}{1 - \frac{R_2 C_{eff_2}}{S_1}(1 - e^{-\frac{S_1}{R_2 C_{eff_2}}})}.$$

Similarly, we find the delay at node 3, i.e., T_{0-3}, and its slew S_3. We continue this delay and slew propagation towards the sinks until we have computed the initial delays and slews at each node in the RC-tree.

After the forward delay and slew propagation phase, based on these slew values we recalculate the new effective capacitance values (using equation 2 for each node by propagating them backward from sinks towards sources (i.e., in order 7, 6, 5, 4, 3, 2, 1, 0). For the sink node, the effective capacitance is always same as the node capacitance.

$C_{eff_7} = C_7$
$C_{eff_6} = C_6$
$C_{eff_5} = C_5$
$C_{eff_4} = C_4 + K_7 * C_{tot_7}$
$C_{eff_3} = C_3 + K_5 * C_{tot_5} + K_6 * C_{tot_6}$
$C_{eff_2} = C_2 + K_4 * C_{tot_4}$
$C_{eff_1} = C_1 + K_2 * C_{tot_2} + K_3 * C_{tot_3}$
$C_{eff_0} = C_0 + K_1 * C_{tot_1}$

where shielding factor K_j denotes the capacitance shielding factor for node j and is given as (equation 2) :

$$K_j = 1 - \frac{2R_j C_{eff_j}}{S_i}(1 - e^{-\frac{S_i}{2R_j C_{eff_j}}})$$

where, S_i denotes the slew at node j's parent, node i.

We now recompute the slew at source node 0, i.e., S_0, using the newly calculated effective capacitance at source node 0, i.e., C_{eff_0} from the driver slew equation, i.e., slew $S_0 = f(C_{eff_0})$.

The process of forward propagation of delay and slew; and backward propagation of effective capacitances is iterated until the slew value at source node differs by less than a user defined threshold (practically set to 1%) from the previous slew value at which time, we have obtained the final interconnect delays, effective source capacitance and source slew.

As stated before, experimental results with interconnects in all the large industrial ASICs, as well as randomly generated RC-lines show that the source slew converges within 1% of the previous slew in less than 5 iterations. It is obvious from the iterative nature of our approach that its computational complexity is only a constant times the computational complexity of the elmore delay model, which is linear in the number of nodes n in the RC-tree. Thus, the computational complexity of our approach is $\mathbf{O}(c.n)$ where c is the number of iterations it takes for the source slew to converge. As stated, this constant has been experimentally observed to be always less than 5.

In the following section, we demonstrate the accuracy and computational efficiency of the iterative refinement delay method using practical industrial ASICs as well as randomly generated RC-lines.

3. EXPERIMENTAL RESULTS

We performed two set of experiments to demonstrate the accuracy and run-time advantage of our iterative refinement approach for computing interconnect delays. In the first set

Table 1: Iterative Refinement Delay Comparison w.r.t AS/X for each node in the 15 segment RC line

Node	Average Ratio	Minimum Ratio	Maximum Ratio
0	2.03	1.51	2.41
1	1.95	1.47	2.30
2	1.85	1.42	2.15
3	1.73	1.38	1.96
4	1.60	1.33	1.75
5	1.47	1.29	1.55
6	1.36	1.25	1.40
7	1.26	1.21	1.29
8	1.18	1.12	1.21
9	1.12	1.04	1.16
10	1.07	0.99	1.13
11	1.04	0.95	1.11
12	1.02	0.93	1.09
13	1.00	0.91	1.08
14	0.99	0.89	1.07
15	0.98	0.88	1.06
Source Slew	0.97	0.86	1.13

of experiments, we compare the accuracy of our iterative refinement approach w.r.t true delays as measured by AS/X[1] simulator [1] on randomly generated RC-lines. In the second set of experiments, to demonstrate the practical effectiveness of our approach, we employed our iterative refinement delay calculator in IBM's physical synthesis tool known as Placement Driven Synthesis (PDS) [3]. These results are given as follows.

3.1 Results on randomly generated RC lines

The accuracy of any interconnect delay model can be evaluated by comparing it to true delays obtained using a device level simulator such as SPICE or AS/X [1]. We generated 10 random instances of a 15 segment RC-line driven by a inverter with $20\mu m$ PMOS and $10\mu m$ NMOS in IBM's $0.12\mu m$ L_{eff} CMOS technology. The input to the inverter is a signal with a slew of 100ps. We simulated these instances with AS/X simulator and also computed the delay at each node with iterative refinement method. Then we computed the ratio of iterative refinement delay to the AS/X delay at each node. Also, at each node, we computed an average, a maximum, and a minimum ratio over all the randomly generated RC-lines.

Table 1 enumerates the results of our experiments. Column 1 gives the node number in the RC-line with node 0 being the source of the RC-line, i.e., output of the driver and node 15 being the farthest end of this 15 segment RC line. Column 2, Column 3, and Column 4 give the average ratio, the minimum ratio, and the maximum ratio of the iterative refinement delay to the AS/X delay for all randomly generated RC-lines. It is clear that on an average the iterative refinement method can achieve highly accurate delays at the far-end of the line (closer to end nodes). The minimum and maximum deviation from true delays is also within reasonable bounds. In addition, the delays at difficult to match source end of the RC-line are also very accurate considering that fact that source end delays are smaller and thus a small variation from true AS/X delays results in large percentage variation. Our results show an improvement over the ECM delay metrics results proposed by Kashyap et. al. (Table 2 in [8]). In fact, our results are significantly better for the more difficult to match source end nodes. The task of an interconnect delay calculator is not only to provide accurate interconnect delays but to also provide accurate effective capacitance load to the CMOS drivers. This is measured by the accuracy of the driver output slew (i.e., source slew) w.r.t AS/X. This information is given in last row of Table 1 where it can be seen that on an average the source slew given by the iterative refinement approach match very closely with AS/X results.

[1] AS/X is IBM's electrical-level simulator similar to SPICE.

3.2 Results on large industrial ASICs with physical synthesis

The real impact of any delay metrics in terms of its run-time advantage and accuracy can be practically evaluated by using it in a design tool that is timing-analysis intensive. For the evaluation of interconnect delay metrics, physical synthesis tool is such an application. In physical synthesis environment, the physical location as well as size and number of various circuits is constantly changing. Thus, the interconnect topology as well as driver size of various nets is also changing. These changing net topologies require several hundred thousand evaluations of interconnect delays during each iteration of placement and synthesis interaction. We employed our iterative refinement delay calculator in IBM's physical synthesis tool known as PDS [3] to demonstrate its practical advantage. The initial release of PDS employed lumped delay model due to its simplicity and concerns regarding excessive run time for physical synthesis of large industrial ASICs if a more complex delay model is used.

We integrated our iterative refinement delay calculator in PDS environment and synthesized several large industrial ASICs. The results are given in Table 2 and are compared with the results when a lumped delay model is used (the previous delay calculator in PDS). Column 1 gives the design name; Column 2 gives the number of gates in the design; Column 3, and Column 4 give the worst optimized slack (ns)[2]; Column 5, and Column 6 give the Figure of Merit (ns); Column 7, and Column 8 give the number of negative data setups; Column 9, and Column 10 give the total run-time in seconds. From Column 3 to Column 10, even columns report the results when lumped delay model is used in comparison to odd columns which report the results of iterative refinement delay calculator. Figure of merit is an integer that gives the cumulative slack of all negative slack circuits in the design and measures the quality of optimized design.

As can be seen from the worst slack as well as figure of merit, and the number of negative setups, lumped delay model significantly overestimates the delays due to pessimistic interconnect delays and the absence of resistive-shielding. It is clear from the results that the use of a more accurate iterative refinement delay model removes the pessimism in interconnect and driver delay timing. In addition, reducing timing pessimism also helps the physical synthesis close on the timing earlier, thereby helping the timing-closure problem. This leads to a reduction in total run-time

[2] Slack calculation in PDS is performed using IBM's internal static timing analysis tool, Einstimer [4].

Table 2: Delay and Run-time comparison of lumped delay and iterative refinement model during physical synthesis.

Design Name	Number of Gates	Worst final Slack slack (ns)		Figure of Merit (ns)		# of Negative data setups		Total CPU run time (sec)	
		Lumped	Iterative	Lumped	Iterative	Lumped	Iterative	Lumped	Iterative
PPC405B3V2	41317	-0.18	-0.18	-31	-29	568	493	4836	4495
CREAM-0	67917	-0.61	-0.55	-716	-614	2000	1885	5088	4563
CHIP-TOP	72144	-1.01	-1.02	-228	-212	486	467	3653	3635
AXQ-TOP	146642	-0.92	-0.39	-1514	-175	6213	2052	10673	10984
APG	210217	-2.97	-2.42	-8465	-5736	9359	6619	11526	11893
IPC-TOP	265691	-1.90	-1.43	-34144	-20219	36994	32256	27016	22734
COBALT	318220	-4.25	-2.35	-10383	-3427	12846	5900	27774	26797
HERC-TOP	444472	-0.29	-0.02	-63	0	513	8	33074	27839

for almost all the designs, which goes against the initial intution that selecting a more accurate delay model results in increased run-time. In general, for all the ASICs listed in Table 2, we obtained a reduction in run time from a total of 123640 seconds to 112942 seconds, i.e., a reduction of 8.7%. This run time advantage depends on the number of negative data setups, i.e., the size of the critical region, i.e, a larger critical region introduces more pessimism with lumped delay model and yields a higher run-time benefit with the new iterative approach. Due to its run-time advantage, and its accuracy of delay modeling, the iterative refinement approach is now the default delay model in IBM's Placement Driven Synthesis tool, PDS.

4. CONCLUSIONS

In this paper, we proposed a new computationally efficient and accurate delay model for estimating interconnect delays. This new estimation method considers the effect of slew as well as resistive shielding of capacitance to yield more accurate delays for both interconnects and driver gate. To account for interdependence of slew and effective capacitance, we iteratively refine the source slew to yield accurate node delays. In addition, it is computationally similar in efficiency to the elmore delay model. Experimental results show that the proposed approach gives not only highly accurate results for far end RC-line delays but also compares very favorably to more difficult to match source end delays and source slews. In addition, use of the proposed delay model in physical synthesis yields significant performance improvement on several large industrial ASICs.

5. REFERENCES

[1] AS/X User's Guide, IBM Corp., 1996.

[2] J. Cong, L. He, C. K. Koh, and P. H. Madden. Performance Optimization of VLSI Interconnect Layout. *Integration: VLSI Journal*, 21:1–94, 1996.

[3] W. Donath, P. Kudva, L. Stok, P. Villarrubia, L. Reddy, A. Sullivan, and K. Chakraborty. Transformational Placement and Synthesis. In *DATE*, pages 194–201, 2000.

[4] Einstimer User's Guide, IBM Corp., 2001.

[5] W. C. Elmore. The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. *Journal of Applied Physics*, 19(1):55–63, 1948.

[6] A. B. Kahng and S. Muddu. Two-Pole Analysis of Interconnect Trees. In *Multi-Chip Module Conference*, pages 105–110, 1995.

[7] A. B. Kahng and S. Muddu. An Analytical Delay model for RLC Interconnects. *IEEE Trans. on CAD*, 16(12):1507–1514, 1997.

[8] C. V. Kashyap, C. J. Alpert, and A. Devgan. An Effective Capacitance based Delay Metric for RC Interconnect. In *ICCAD*, pages 229–234, 2000.

[9] R. Kay and L. Pileggi. PRIMO: Probability Interpretation of Moments for Delay Calculation. In *DAC*, pages 463–468, 1998.

[10] R. Puri, D. S. Kung, and A. D. Drumm. System and Method for Fast Interconnect Delay Estimation through Iterative Refinement, US patent application, (filed) September, 2000.

[11] J. Qian, S. Pullela, and L. Pillage. Modeling the Effective Capacitance for the RC Interconnects of CMOS Gates. *IEEE Trans. on CAD*, 13(12):1526–1535, 1994.

[12] C. L. Ratzlaff, S. Pullela, and L. Pillage. Modeling the RC-Interconnect Effects in a Hierarchical Timing Analyzer. In *CICC*, pages 15.6.1–15.6.4, 1992.

[13] J. Rubinstein, P. Penfield, and M. A. Horowitz. Signal Delay in RC Tree Networks. *IEEE Trans. on CAD*, 2(3):202–210, 1983.

[14] B. Tutuianu, F. Dartu, and L. Pileggi. Explicit RC-Circuit Delay Approximation Based on the First Three Moments of the Impulse Response. In *DAC*, pages 611–616, 1996.